Software Framework updates

Jin Huang (BNL)

In discussion with Chris Pinkenburg (BNL), Michael P. McCumber (LANL) and may others

PHENIX

Overview

- We are working on a few items improving the sPHENIX framework. Introduced today:
  - Repository management
  - New format for PHG4Hit
  - Some of more items on the list
- These are proposals for now, suggestions are welcomed
- If agreed upon, we are trying to formalize them into policies and will be introduced in the tutorials in the July sPHENIX workfest: https://indico.bnl.gov/conferenceDisplay.py?confId=1237

Repository Management

sPHENIX software on GitHub

- GitHub is largest online code hoster in the world.
- Chris moved sPHENIX software to GitHub: https://github.com/sPHENIX-Collaboration
- Three teams established with repository write privilege:
  - **Collaborators** : write the analysis modules, propose change to core software, **everyone invited**
  - **coresoftware-developpers** : edit or approve pull request for nightly-built packages.
  - **Owner:** administrative works (add member, etc.)

Repository	Purpose	PHENIX CVS equ.	Write permission team
coresoftware	Nightly build core software. Most strict quality control	offline/package offline/framework simulation/g4simulation	**coresoftware-developpers**
macros	Standard macro to run production	simulation/g4simulation/macros offline/data_production	**coresoftware-developpers**
analysis	Analysis modules	offline/analysis offline/AnalysisTrain/	**Collaborators**
calibrations	Tmp. place for calibrations	Database server	**coresoftware-developpers**
utilities	Maintenance toolkit on RCF	utils	**coresoftware-developpers**
tutorials	Official tutorial and test	N/A	**coresoftware-developpers**

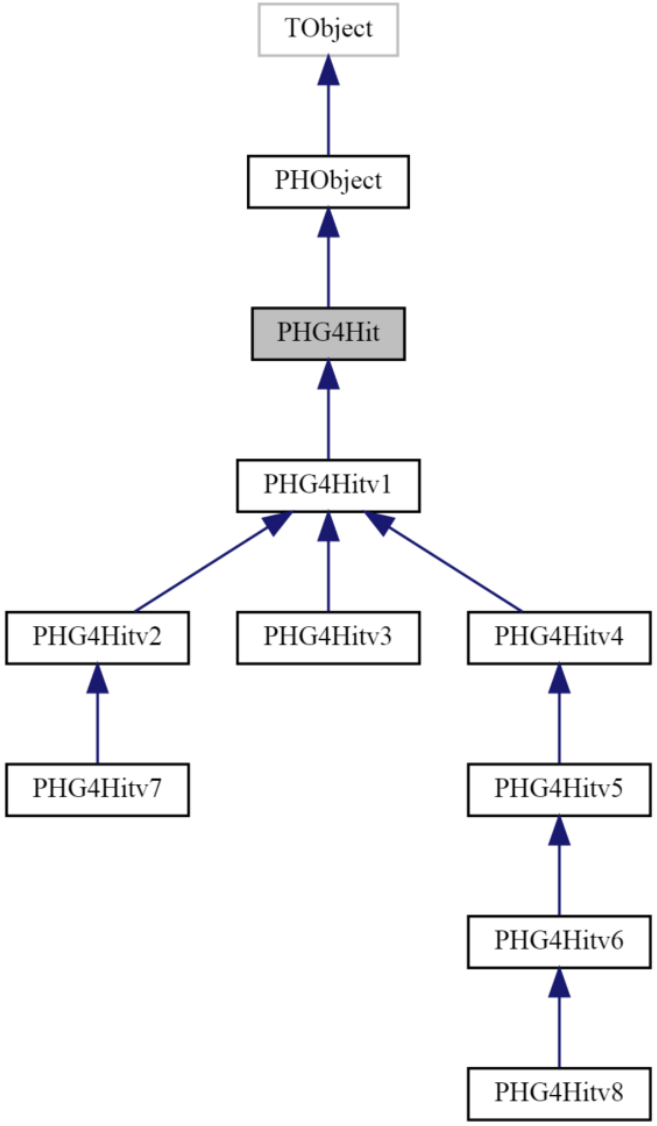
Collaborative editing on GitHub

- GitHub support two types of collaborative editing, we would use both for different repository
- For nightly-built software
  - Software package managers can directly update the code:
    - e.g. https://github.com/sPHENIX-Collaboration/coresoftware/commit/680f014c35751db31a111fd99039e8690d94d2f5
  - Collaborators can folk the repository -> make modification -> test -> send a folk request. Examples:
    - https://github.com/sPHENIX-Collaboration/coresoftware/pull/3
    - https://github.com/sPHENIX-Collaboration/coresoftware/pull/7
- For analysis modules
  - Every collaborator we free to add and edit modules
  - e.g. https://github.com/sPHENIX-Collaboration/analysis/commit/ace66a7c5a5981c2300ee87d8a3ba0d2eb96d684

Solving the many versions of PHG4Hits

What's going on with PHG4Hit

- PHG4Hits are the DST data object we export raw Geant4 information
  - Current the only bridge for detector to save hit information and pass down to
- Difference of subsystems require distinctive additional marking on PHG4Hit (e.g. index of SVX/Cal., store path length of particle)
- Current solution is to develop many versions of PHG4Hit and they keep growing



Proposed solution

- Proposed commitment:
  - Summary: https://github.com/sPHENIX-Collaboration/coresoftware/pull/7
  - PHG4Hit: https://github.com/sPHENIX-Collaboration/coresoftware/pull/7/files#diff-069fdb2e441c8660d1323b7440b58dc1
  - PHG4Hitv1: https://github.com/sPHENIX-Collaboration/coresoftware/pull/7/files#diff-0f9943da181bcbd453d56d4beb262202
- Keep the core part of PHG4Hit in version 1
- Allow PHG4Hit to be tagged by a variable set of properties
  - Each labelled by a ID, [1-255], with an enum name
  - Each property can be float, int or uint value
  - Different subsystem (or different part within a subsystem) can produce hits with different sets of property
  - Get set function of previous version is still supported

Currently supported properties

```
//! add a short name to PHG4Hit::get_property_name
enum PROPERTY {//

  //-- hit properties: 1 - 10  --
  //! ionizing energy loss
  prop_eion = 1,

  //! for scintillation detectors, the amount of light produced
  prop_light_yield = 2,

  //-- event properties: 10 - 20  --

  //! momentum
  prop_px = 10,
  prop_py ,
  prop_pz ,

  //! pathlength
  prop_path_length = 15,

  //-- detector specific IDs: 100+ --

  //! layer ID
  prop_layer = 101,
  //! scintillator ID
  prop_scint_id = 102,

  //! SVX stuff
  prop_strip_z_index = 110,
  prop_strip_y_index,
  prop_ladder_z_index,
  prop_ladder_phi_index,

  //! generic indexes
  prop_index_i = 121,
  prop_index_j ,
  prop_index_k ,
  prop_index_l ,



  //! max limit in order to fit into 8 bit unsigned number
  prop_MAX_NUMBER = UCHAR_MAX
};

virtual bool  has_property(PROPERTY prop_id) const {return false;}
virtual float get_property_float(PROPERTY prop_id) const {return NAN;}
virtual int   get_property_int(PROPERTY prop_id) const {return INT_MIN;}
virtual unsigned int   get_property_uint(PROPERTY prop_id) const {return UINT_MAX;}
virtual void  set_property(PROPERTY prop_id, float value) {return;}
virtual void  set_property(PROPERTY prop_id, int value) {return;}
virtual void  set_property(PROPERTY prop_id, unsigned int value) {return;}
static const char * get_property_name(PROPERTY prop_id);
```

Old IO functions are still supported
So changes are transparent

```
virtual float get_px(const int i) const {return  get_property_float(prop_px);}
virtual float get_py(const int i) const {return  get_property_float(prop_py);}
virtual float get_pz(const int i) const {return  get_property_float(prop_pz);}
virtual float get_eion() const          {return  get_property_float(prop_eion);}
virtual float get_light_yield() const   {return  get_property_float(prop_light_yield);}
virtual float    get_path_length() const{return  get_property_float(prop_path_length);}
virtual unsigned int get_layer() const  {return  get_property_uint(prop_layer);}
virtual int get_scint_id() const        {return  get_property_int(prop_scint_id);}
virtual int get_strip_z_index() const   {return  get_property_int(prop_strip_z_index);}
virtual int get_strip_y_index() const   {return  get_property_int(prop_strip_y_index);}
virtual int get_ladder_z_index() const  {return  get_property_int(prop_ladder_phi_index);}
virtual int get_ladder_phi_index() const{return  get_property_int(prop_ladder_phi_index);}
virtual int get_index_i() const{return  get_property_int(prop_index_i);}
virtual int get_index_j() const{return  get_property_int(prop_index_j);}
virtual int get_index_k() const{return  get_property_int(prop_index_k);}
virtual int get_index_l() const{return  get_property_int(prop_index_l);}

virtual void set_px(const int i, const float f) {set_property(prop_px,f);}
virtual void set_py(const int i, const float f) {set_property(prop_py,f);}
virtual void set_pz(const int i, const float f) {set_property(prop_pz,f);}
virtual void set_eion(const float f)            {set_property(prop_eion,f);}
virtual void set_light_yield(float f)           {set_property(prop_light_yield,f);}
virtual void set_path_length(float f)           {set_property(prop_path_length,f);}
virtual void set_layer(const unsigned int i)    {set_property(prop_layer,i);}
virtual void set_scint_id(const int i)          {set_property(prop_scint_id,i);}
virtual void set_strip_z_index(const int i)     {set_property(prop_strip_z_index,i);}
virtual void set_strip_y_index(const int i)     {set_property(prop_strip_y_index,i);}
virtual void set_ladder_z_index(const int i)    {set_property(prop_ladder_phi_index,i);}
virtual void set_ladder_phi_index(const int i)  {set_property(prop_ladder_phi_index,i);}
virtual void set_index_i(const int i)  {set_property(prop_index_i,i);}
virtual void set_index_j(const int i)  {set_property(prop_index_j,i);}
virtual void set_index_k(const int i)  {set_property(prop_index_k,i);}
virtual void set_index_l(const int i)  {set_property(prop_index_l,i);}
```

Test on file size

- Properties tags are saved with a std::map of ID (8bit int) to 32bit numbers.
- Concerns whether ROOT can efficiently save such structure to ROOT files
- Tested with 100 10GeV electron simulation to sPHENIX calorimeters
- Results are: https://github.com/sPHENIX-Collaboration/coresoftware/pull/7
  - Very minor inflation on DST file size (+3%)
  - Reduced memory size for SPACAL hits (-2%) since less property number are needed
  - File size after compressed for SPACAL hits (+4%) are larger due to more complex structure and less compression.
- In summary, not much impact to the size of storage

Other on TODO lists

Other items on the list, help suggestions welcomed

- Support post-production analysis off-site RCF: split DST storage lib and modular lib
- Calibration objects and database connection
- Geometry handling
- ROOT 6 support? Whether and when?
- And more. Help are welcomed!